cs4_section3

February 16, 2019

What's the difference between a shallow and a deep copy?

Answer: A shallow copy only copies a reference *memory address* or the location of a variable. If the values at the memory are changed or the location of the original address is altered, the new copy will also take on the altered value. A deep copy copies the memory address directly. Thus, even if the memory address of the original variable changes, a deep copy will preserve the original values of the variable.

Let's take a look at an example: what is printed out from the following code?

```
def double_var(x):
   ''' function takes in an int x and returns double its value ''' 
    x= x*2
    return x



x = 4
double_var(x)
#What is printed in this code? Why?
print(x)
```

Answer: 4 is printed. This is a question about global vs local scope. Now let's alter the code: what is printed this time?

```
def double_var(x):
   ''' function takes in an int x and returns double its value ''' 
    x= x*2
    return x



x = [4]
double_var(x)
#What is printed in this code? Why?
print(x)
```

Answer: `[4,4]` is printed. This is because a list is mutable, so changing x within 1 function changes the value elsewhere. How would you modify the above code to print the original value of x (*Hint:* Create a deep copy!)

2D lists and functions that operate on them are very important. Try your hand at making a function that has the following behavior: every even-index column should be doubled, and a

deep-copied matrix should be returned. For instance,

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 & 7 \end{bmatrix}$$

becomes

$$\begin{bmatrix} 2 & 2 & 6 & 4 & 10 \\ 6 & 4 & 10 & 6 & 14 \end{bmatrix}$$

```
In [3]: def double_even_column(matrix):
            '''
            Write your answer here
            '''
            return
```

Possible answer: make sure you understand what this code is doing as this is very important to hw3 and many future homeworks.

```
In [4]: def double_even_column(matrix):
            #Create a deep copy
            matrix_copy = [sub[:] for sub in matrix]
            for r in range(len(matrix)):
                for c in range(len(matrix[0])):
                    if c%2 == 0:
                        matrix_copy[r][c]= matrix_copy[r][c]*2
            return matrix_copy
```

Make sure that you are familiar with printing vs returning, printing vs testing (assert statements), indentation, variable types, and commenting. If you are not, please reach out to a TA asap to understand these concepts better.

```
In [ ]:
```